SSCM Exercise 2

Nikolaus Czernin - 11721138

```
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# install.packages("FRACTION")
library("FRACTION")
```

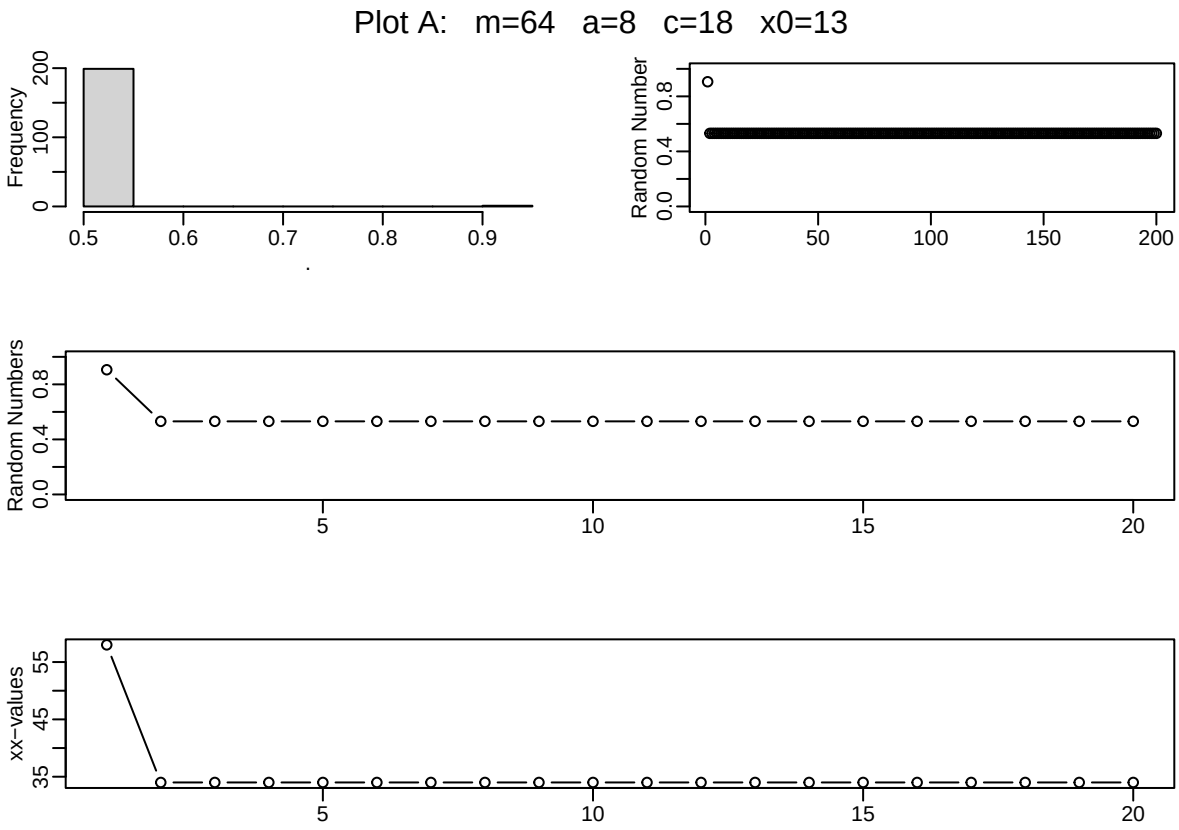
Linear Congruential Random Number Generator

```
# prepare the PRNG function
lcrng <- function(n, m, a, c=0, x0){
  us <- numeric(n)
  # keep an extra vector for the x values
  xs <- numeric(n)
  for (i in 1:n){
    x0 <- (a * x0 + c) %% m
    xs[i] <- x0
    us[i] <- x0 / m
  }
  list("u"=us, "x"=xs)
}


visualize_random_numbers <- function(n, m, a, c, x0, title=""){
  # generate random numbers with given params
  prns <- lcrng(n, m, a, c, x0)
  df <- data.frame(
    i = 1:n,
    u = prns$u,
    x = prns$x,
    n = n,
```

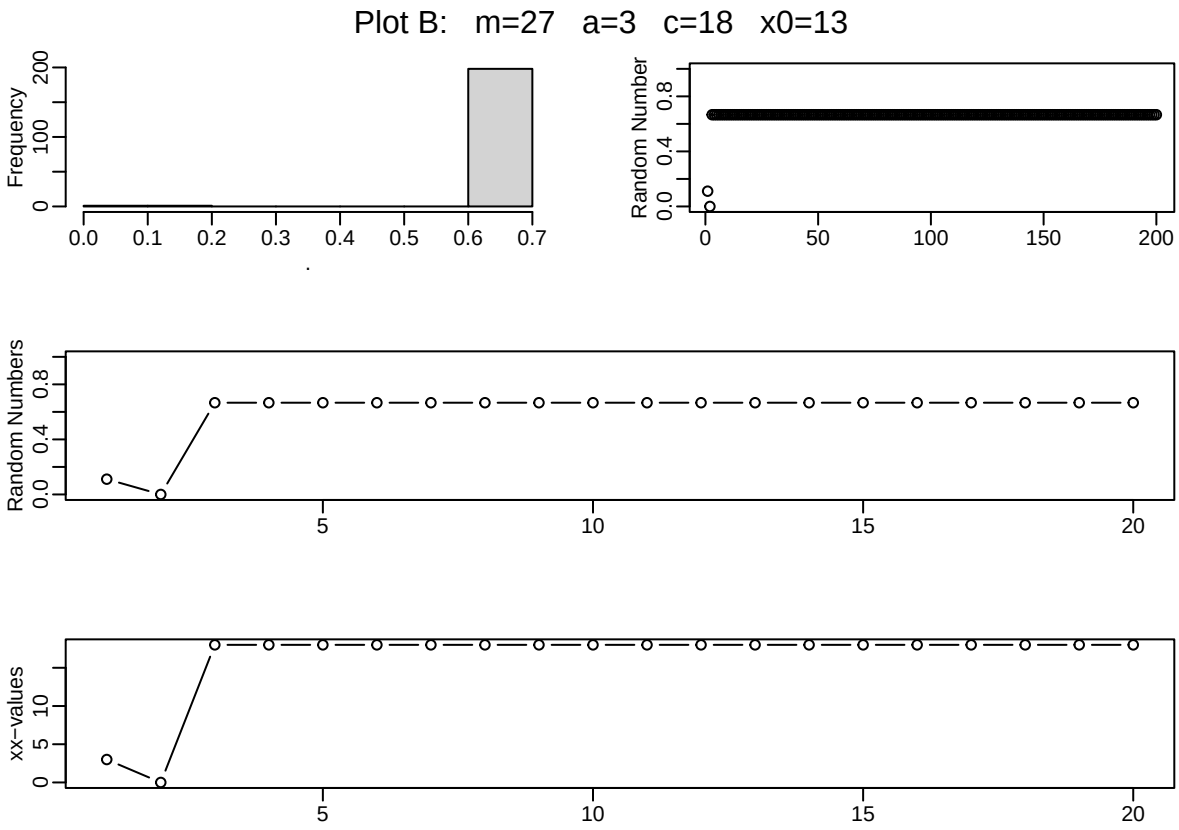
```
    m = m,
    a = a,
    c = c,
    x0 = x0
  )

  # prepare a double-plot window
# Set up the layout: 2 rows, 2 columns
  layout(matrix(c(1, 2, 3, 3, 4, 4), nrow = 3, byrow = TRUE))

  par(mar = c(3, 3.5, 2.5, 2),
      mgp = c(1.5, 0.5, 0))

  # create the plots
  df$u %>% hist(main="")
  plot(df$i, df$u, xlab="", ylab="Random Number", ylim=c(0, 1))

  # for the line plot, only use the first 20 numbers
  df_head <- df %>% head(20)
  plot(df_head$i, df_head$u, type="b", xlab="", ylab="Random Numbers", ylim=c(0, 1))
  first_cycle <- filter(df, x==x0) %>% head(1) %>% .$i
  abline(v=first_cycle, col="blue")

  # plot also the x-values
  plot(df_head$i, df_head$x, type="b", xlab="", ylab="xx-values")
  abline(v=first_cycle, col="blue")

  # make a custom title with the parameters
  text <- paste0(title, ":   m=", m, "   a=", a, "   c=", c, "   x0=", x0, "\n")
  mtext(text, side=3, outer=TRUE, line=-3)

}

# visualize_random_numbers(200, 9, 8, 7, 6, "Plot A")
```

In this chunk, I defined a PRNG function that uses the Linear Congruential Random Number Generator algorithm. It expects all parameters as arguments. I defined another function that visualizes the result using a histogram, a scatterplot and a lineplot of the first values to visualize a sequence-loop. It also visualizes the x-values for better analysis.

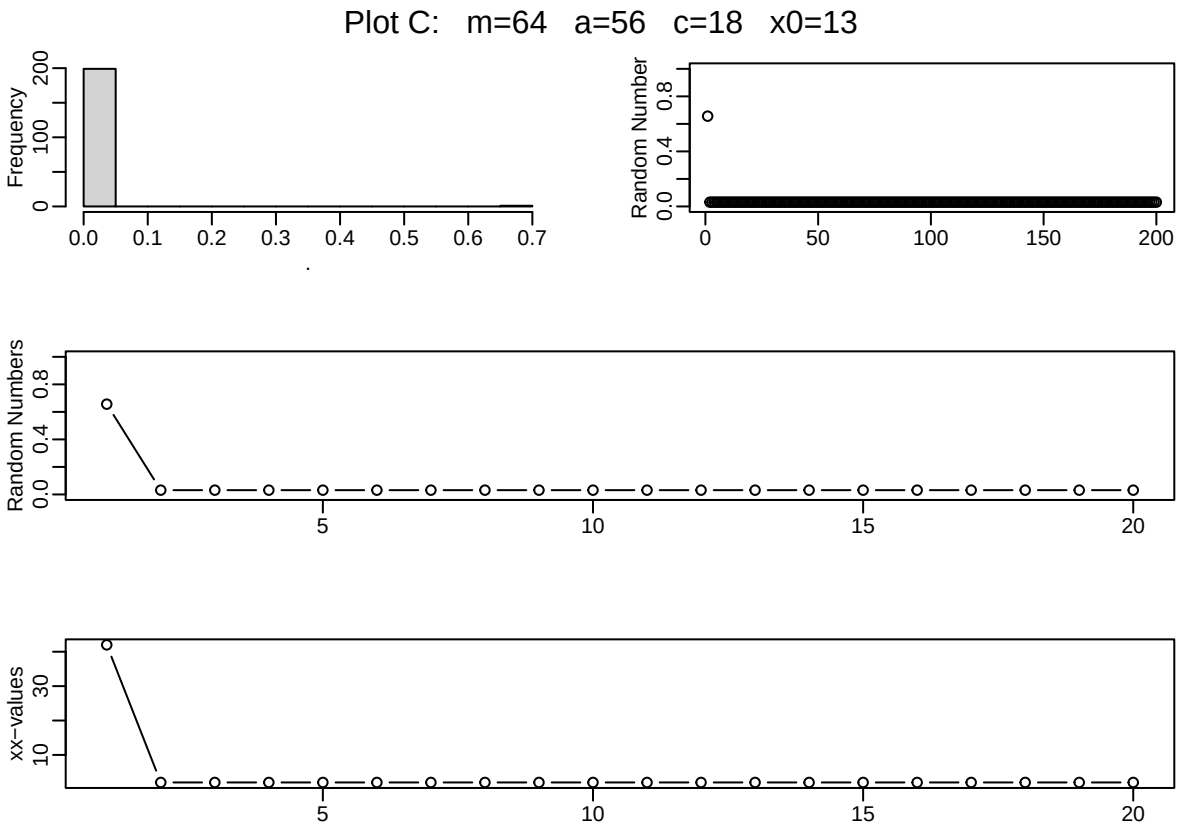
```
visualize_random_numbers(200,  64,   8,  18,  13, "Plot A")
```



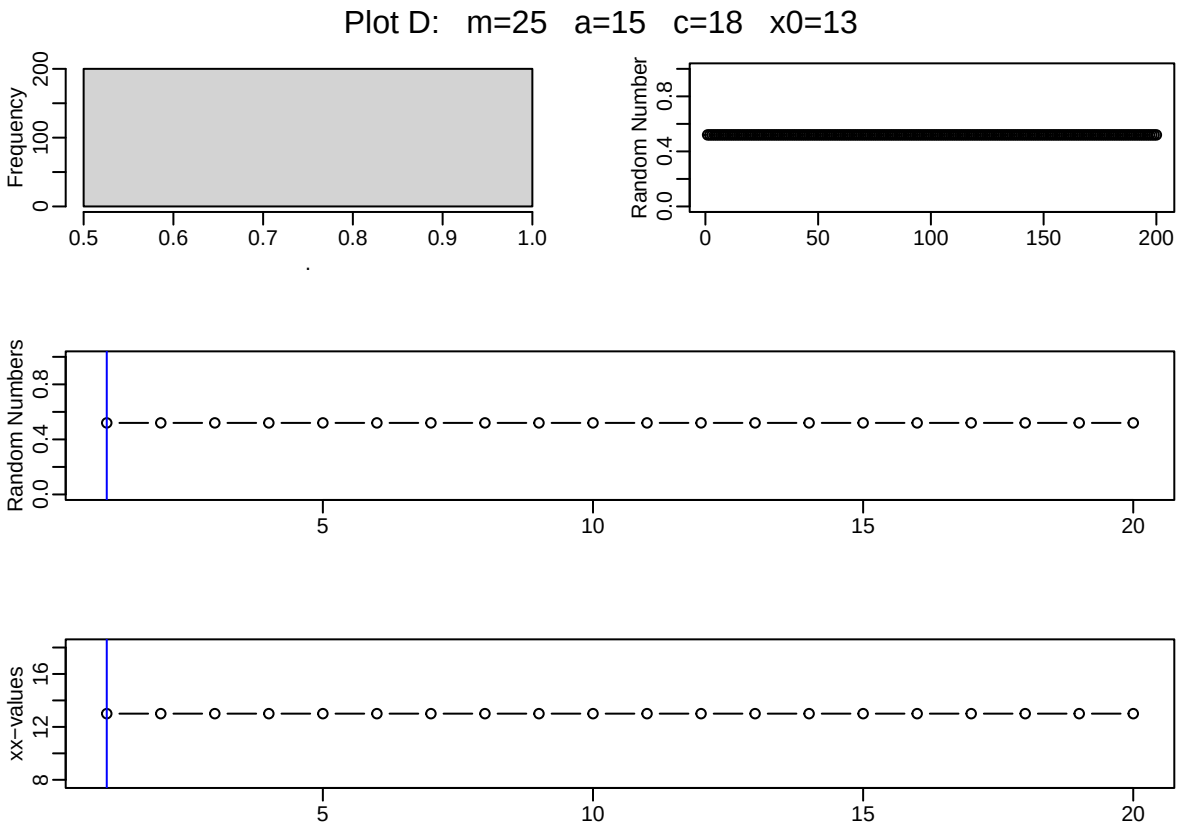

```
visualize_random_numbers(200,  27,   3,  18,  13, "Plot B")
```




```
visualize_random_numbers(200,  64,  56,  18,  13, "Plot C")
```



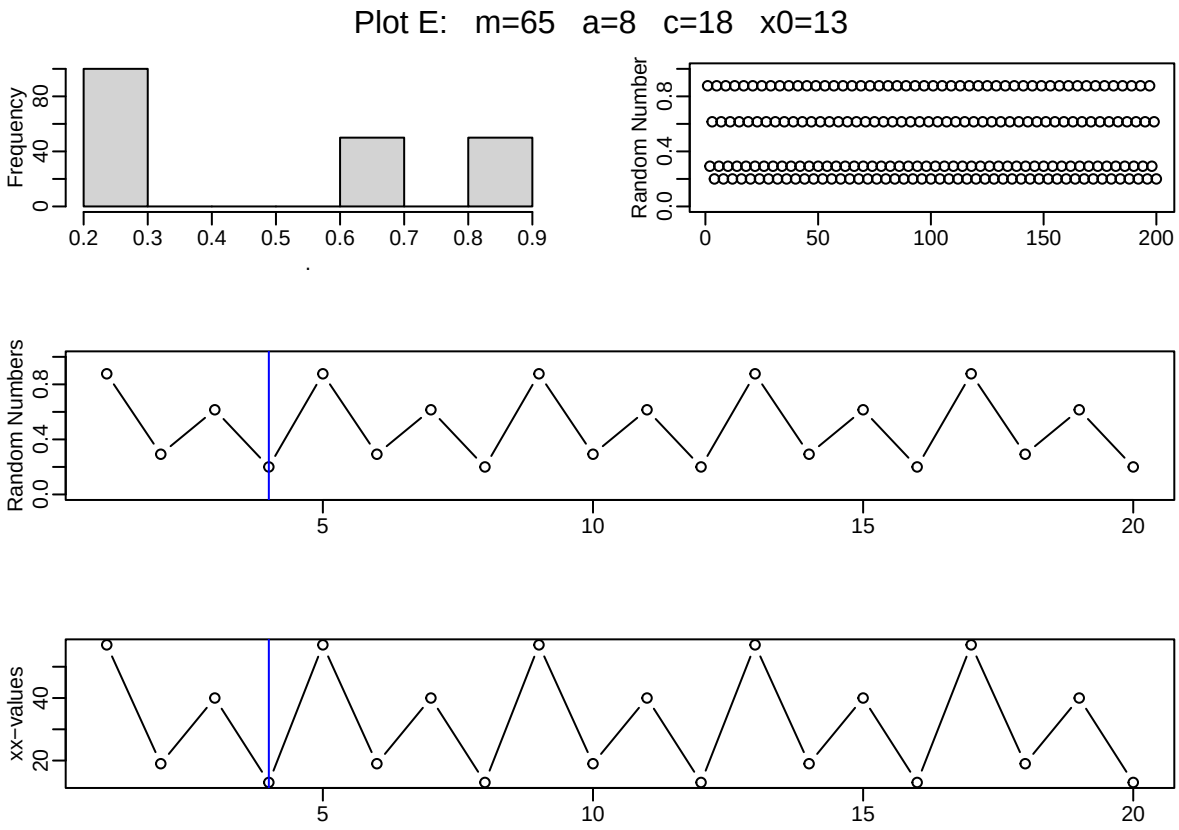

```
visualize_random_numbers(200,  25,  15,  18,  13, "Plot D")
```



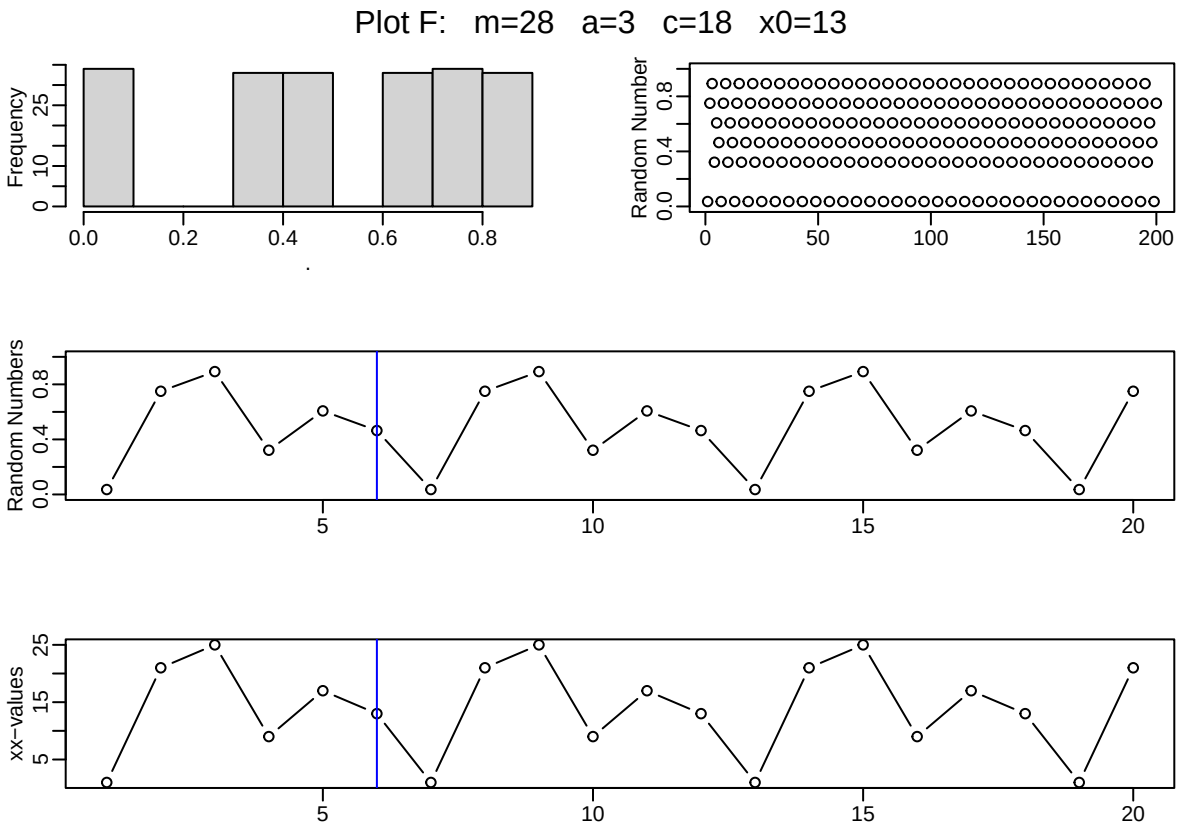
Plots A through D above all loop very early. What they have in common is that m and a have common denominators. After numbers in the sequence, the values stagnate.

The blue lines mark the first iteration, where the generated x-value is equal to the initial x_0 value, which is the latest point of a cycle restarting, though a cycle may start even earlier too.

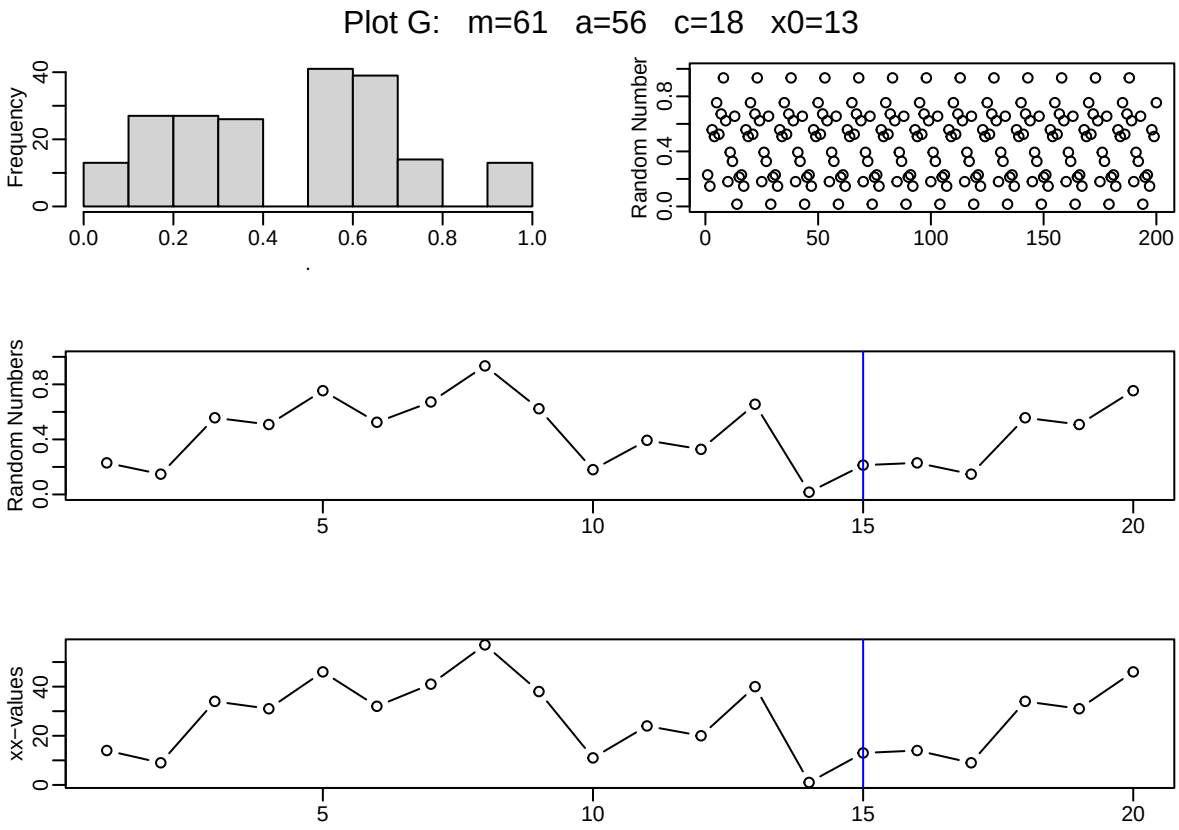
```
visualize_random_numbers(200,  65,   8,  18,  13, "Plot E")
```



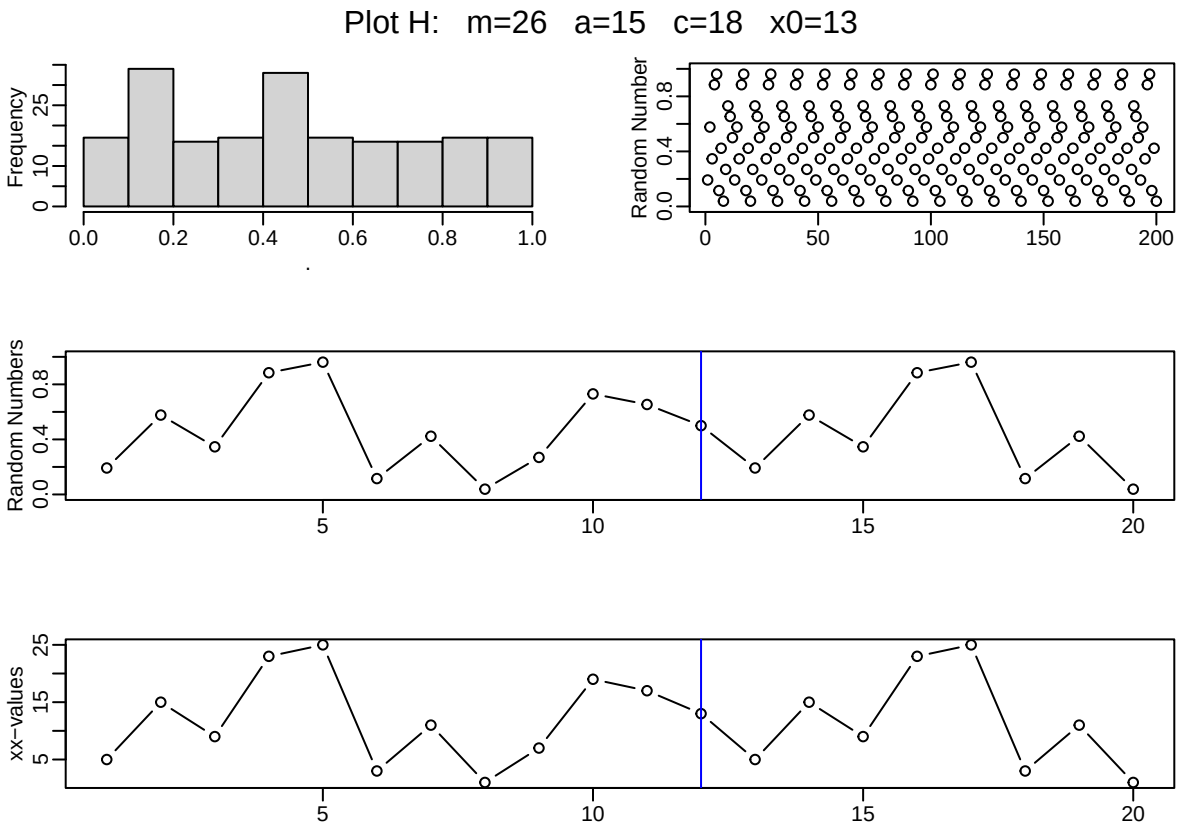

```
visualize_random_numbers(200,  28,   3,  18,  13, "Plot F")
```




```
visualize_random_numbers(200,  61,  56,  18,  13, "Plot G")
```

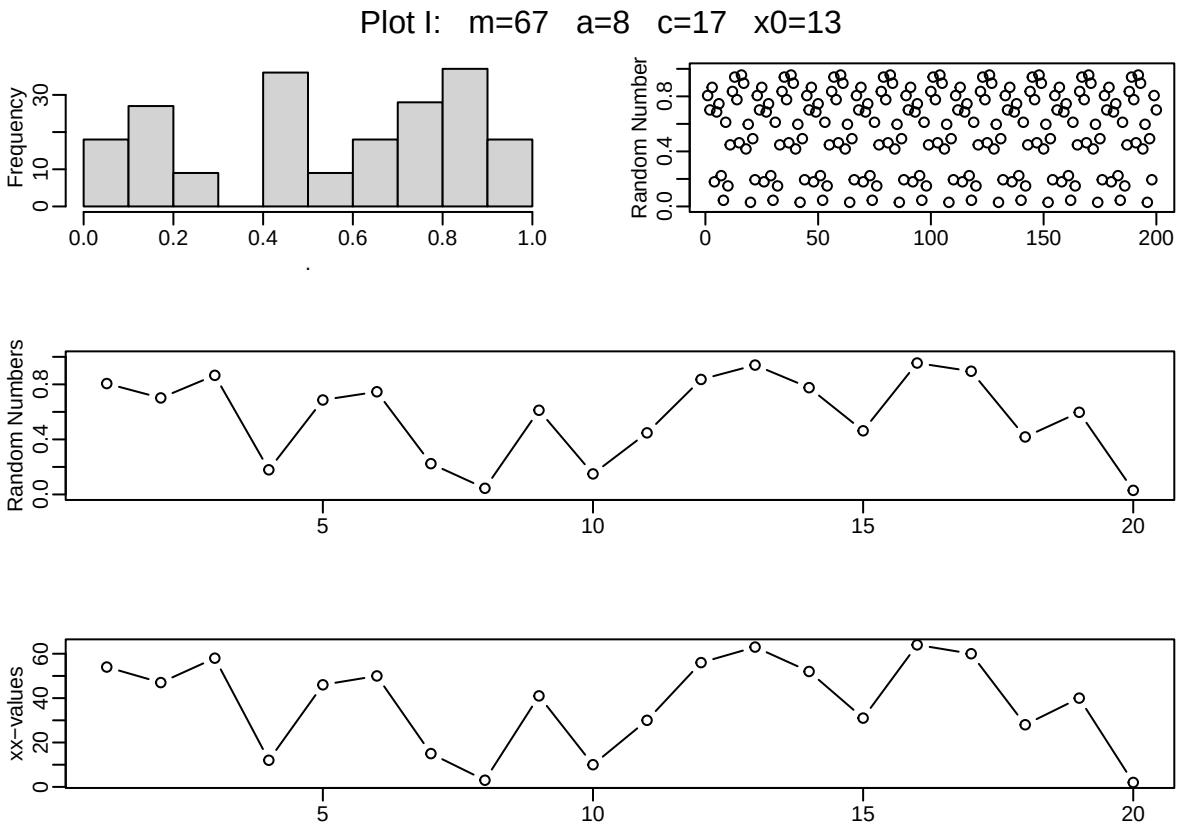



```
visualize_random_numbers(200,  26,  15,  18,  13, "Plot H")
```

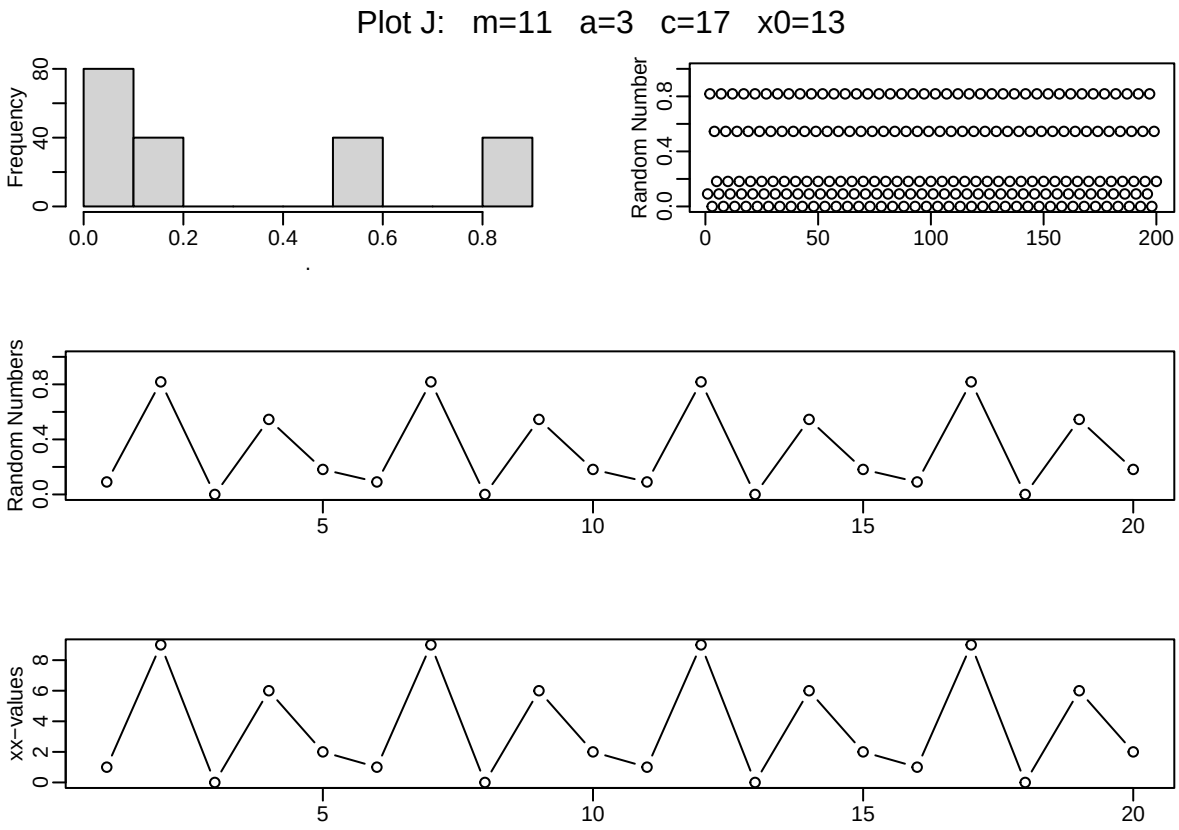


When using m values that have no common denominator with a the loops are created later, as seen in plots E through H.

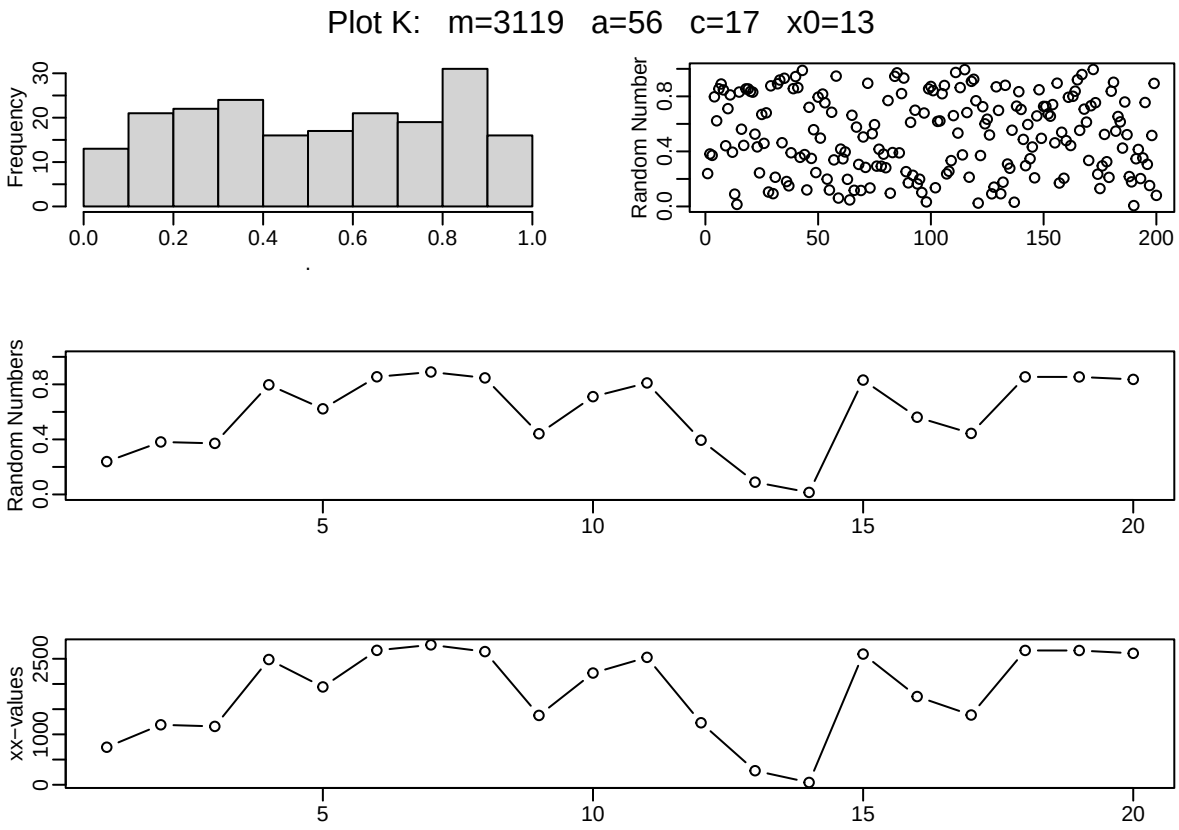
```
visualize_random_numbers(200,  67,   8,  17,  13, "Plot I")
```



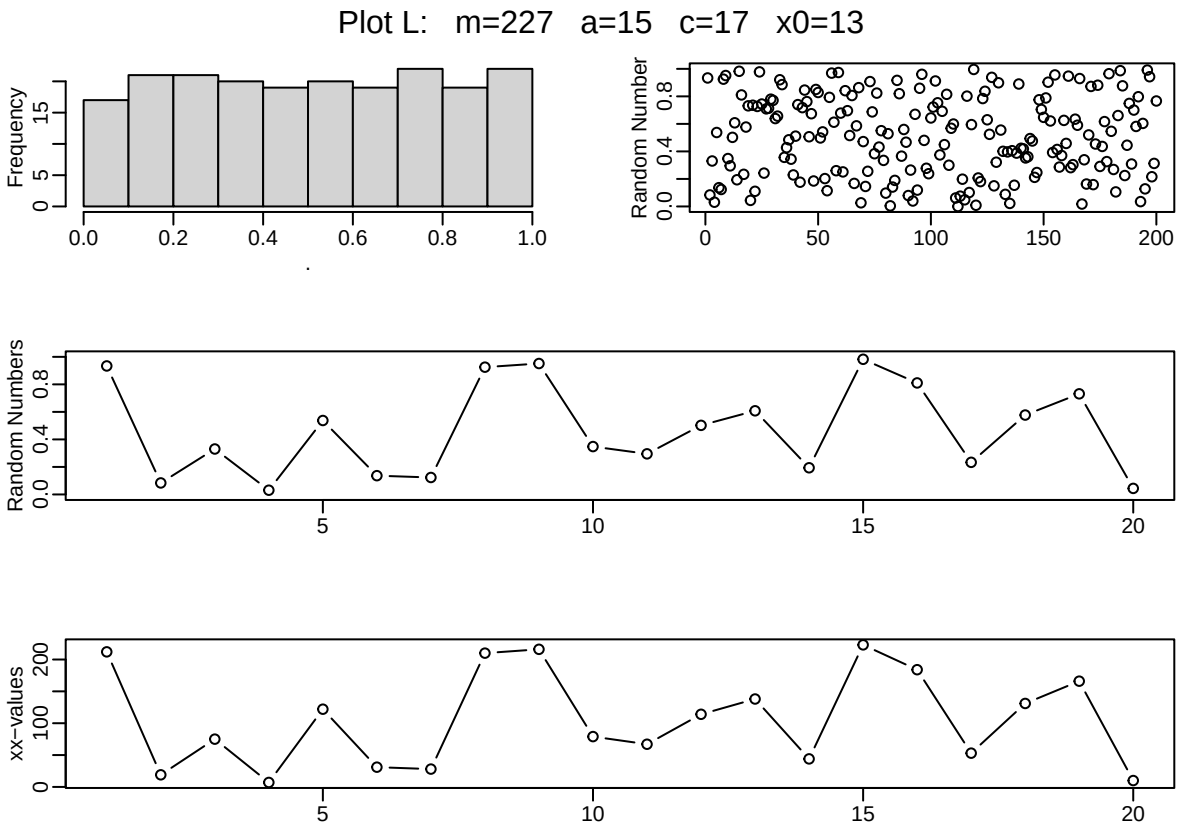

```
visualize_random_numbers(200,  11,   3,  17,  13, "Plot J")
```




```
visualize_random_numbers(200,  3119,  56,  17,  13, "Plot K")
```

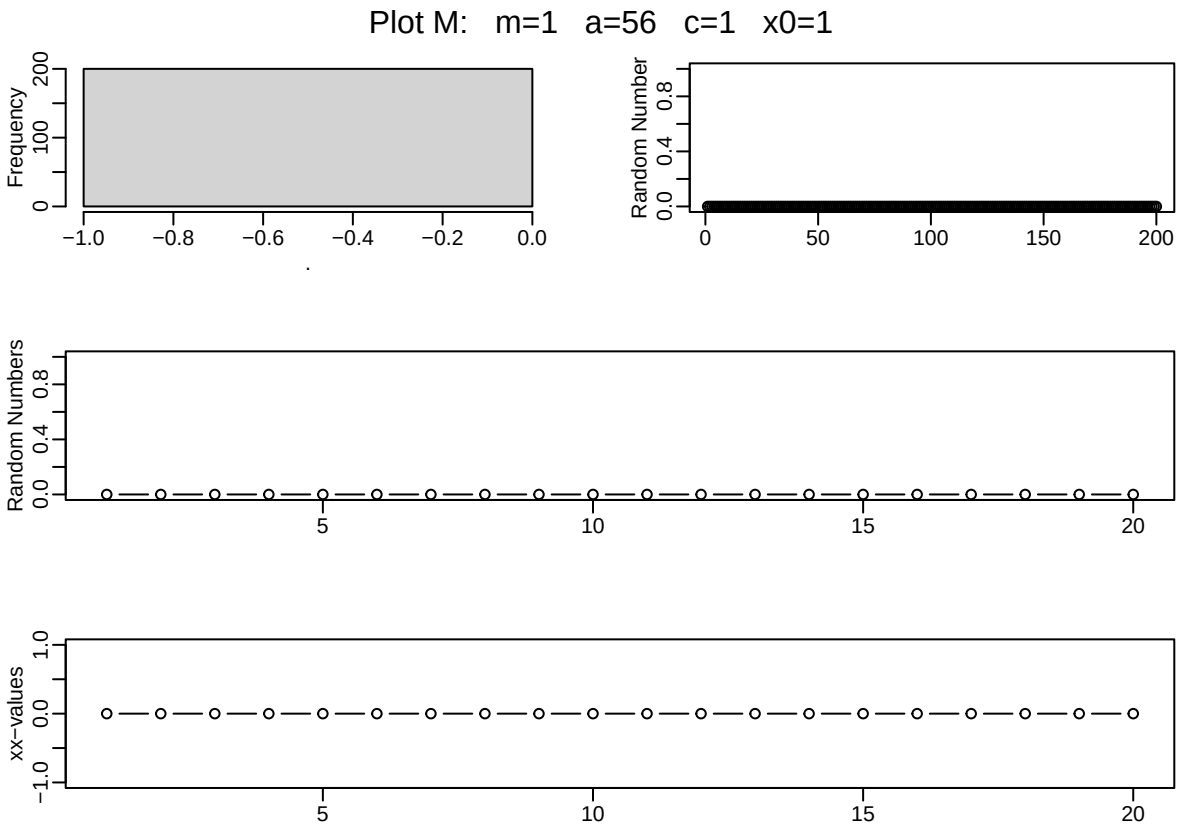



```
visualize_random_numbers(200,  227,  15,  17,  13, "Plot L")
```

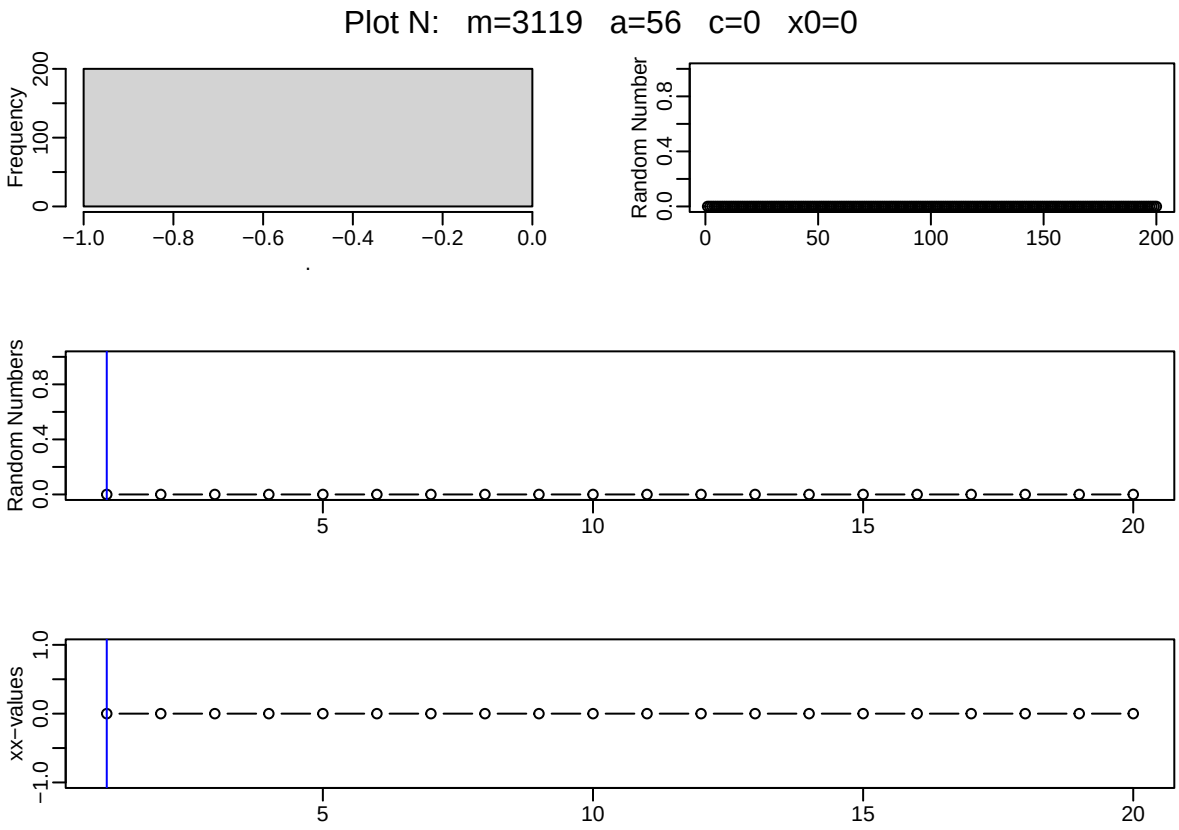


As seen in Plots I through L, when using large prime numbers as m, i.e. 3-digit value or higher, there are no apparent cycles in the random number sequences anymore.

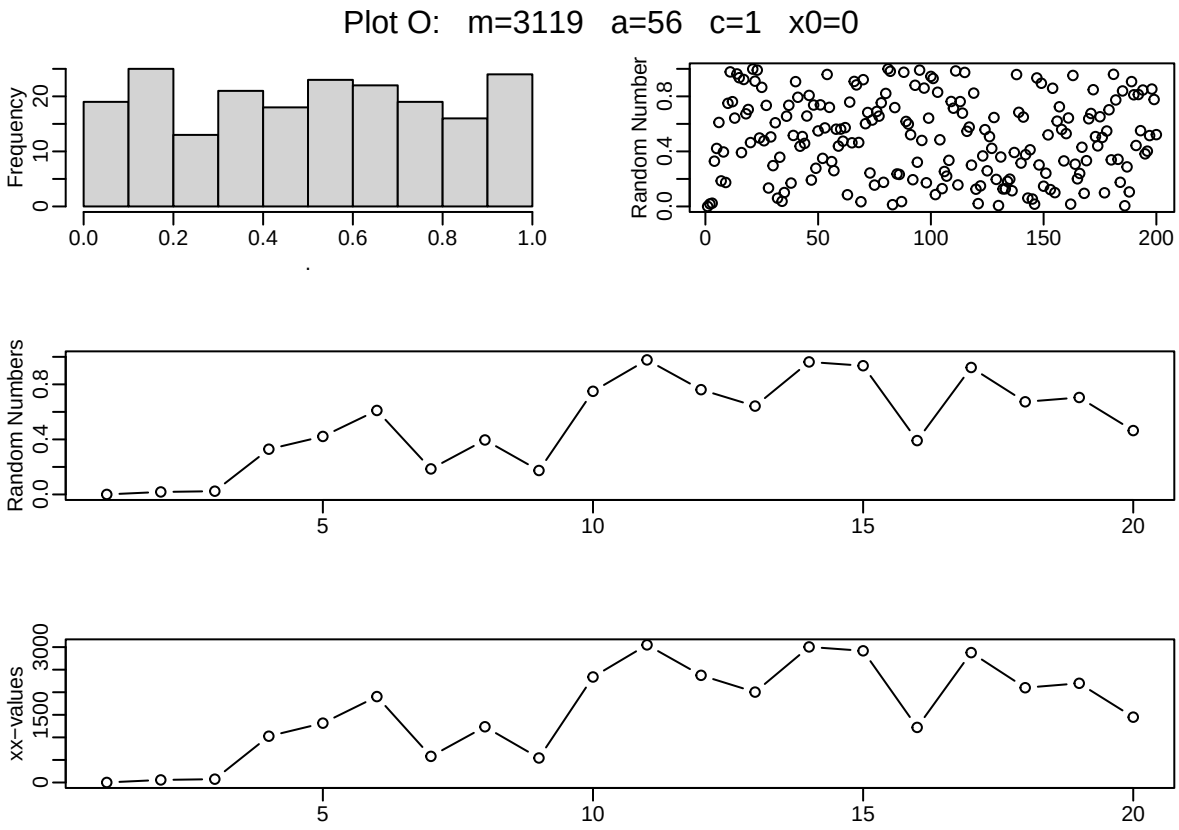
```
visualize_random_numbers(200,  1,  56,  1,  1, "Plot M")
```



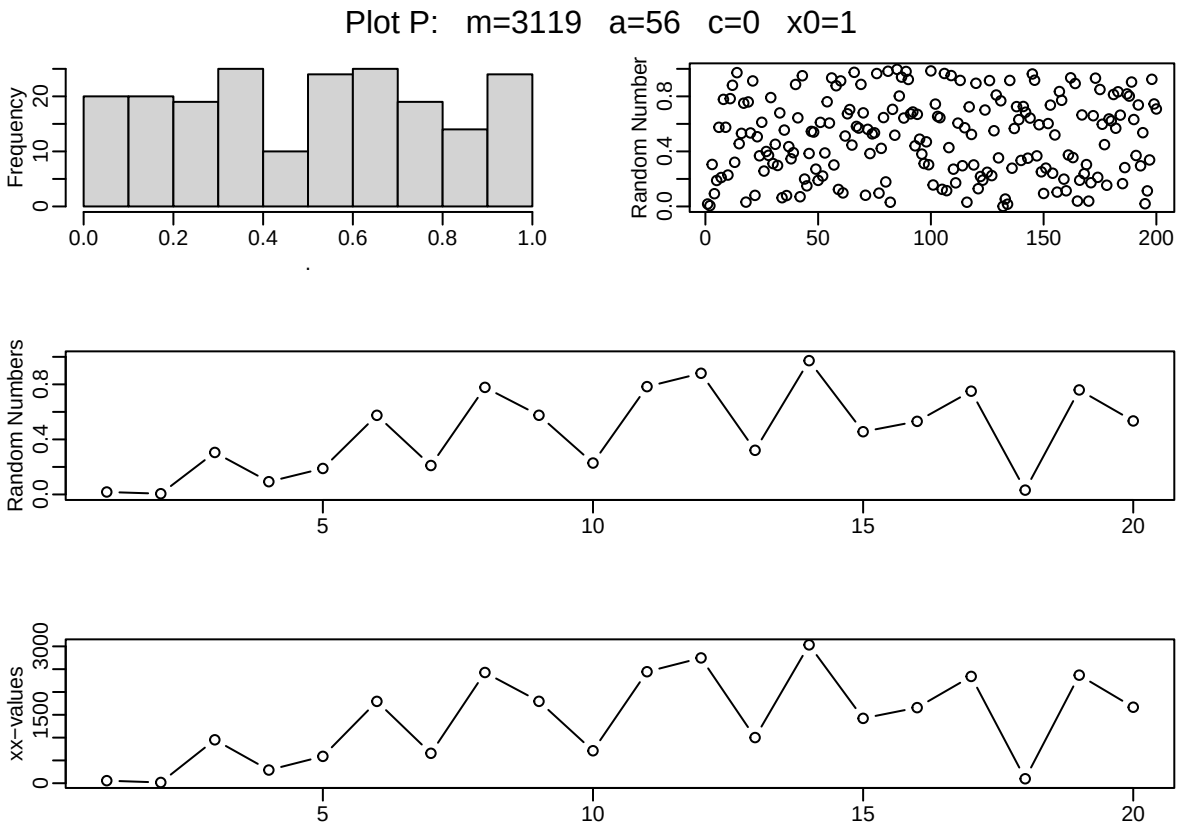

```
visualize_random_numbers(200,  3119,  56,  0,  0, "Plot N")
```




```
visualize_random_numbers(200,  3119,  56,  1,  0, "Plot 0")
```

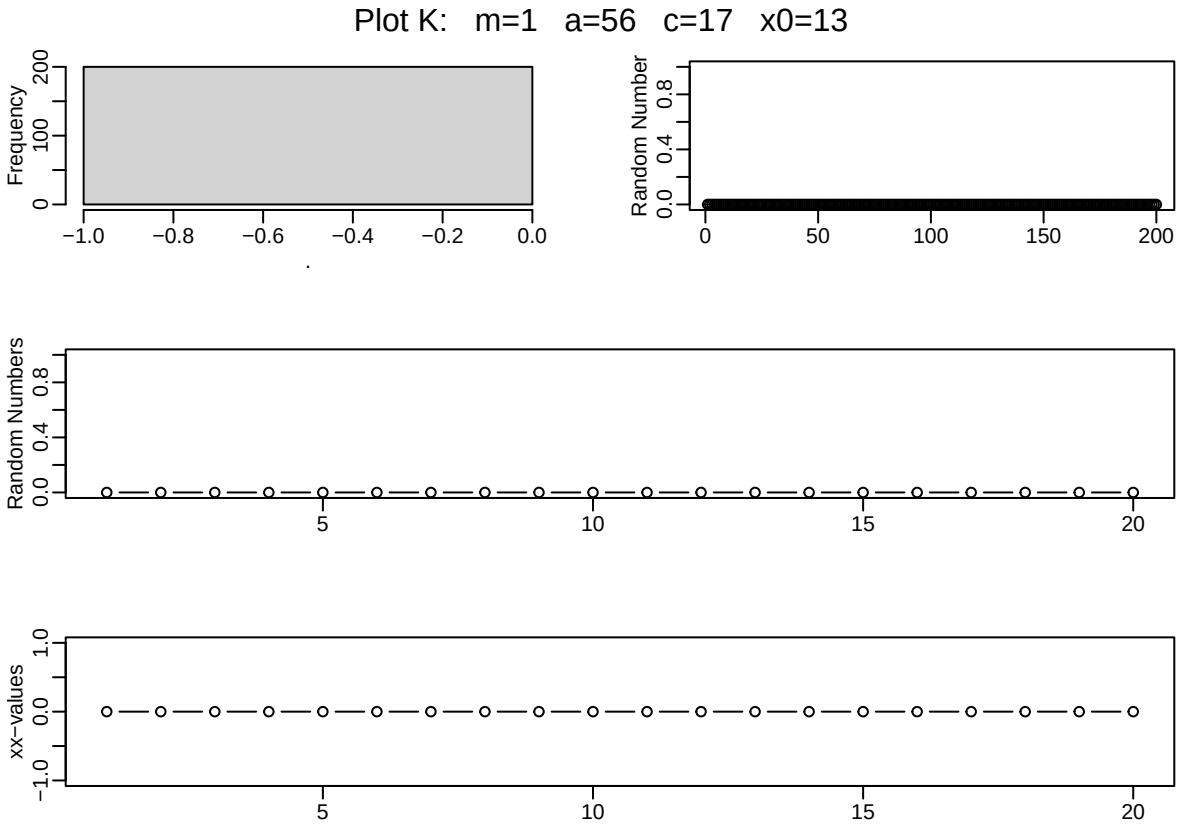



```
visualize_random_numbers(200,  3119,  56,  0,  1, "Plot P")
```



Plot M shows that if m is 1, the random numbers will be all zeros, as modulus of 1 will always be 0 on integers. Plot N shows that having both c and x_0 equal to zero leads to a sequence producing only zeros. If either are unequal to zero, you get a random sequence with seemingly no cycles, as seen in plots O and P.

```
visualize_random_numbers(200,  1,  56,  17,  13, "Plot K")
```


Plot K: m=1 a=56 c=17 x0=13
Frequency
0
100
200
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
Random Number
0.0
0.4
0.8
0
50
100
150
200
Random Numbers
0.0
0.4
0.8
5
10
15
20
xx−values
−1.0
0.0
1.0
5
10
15
20